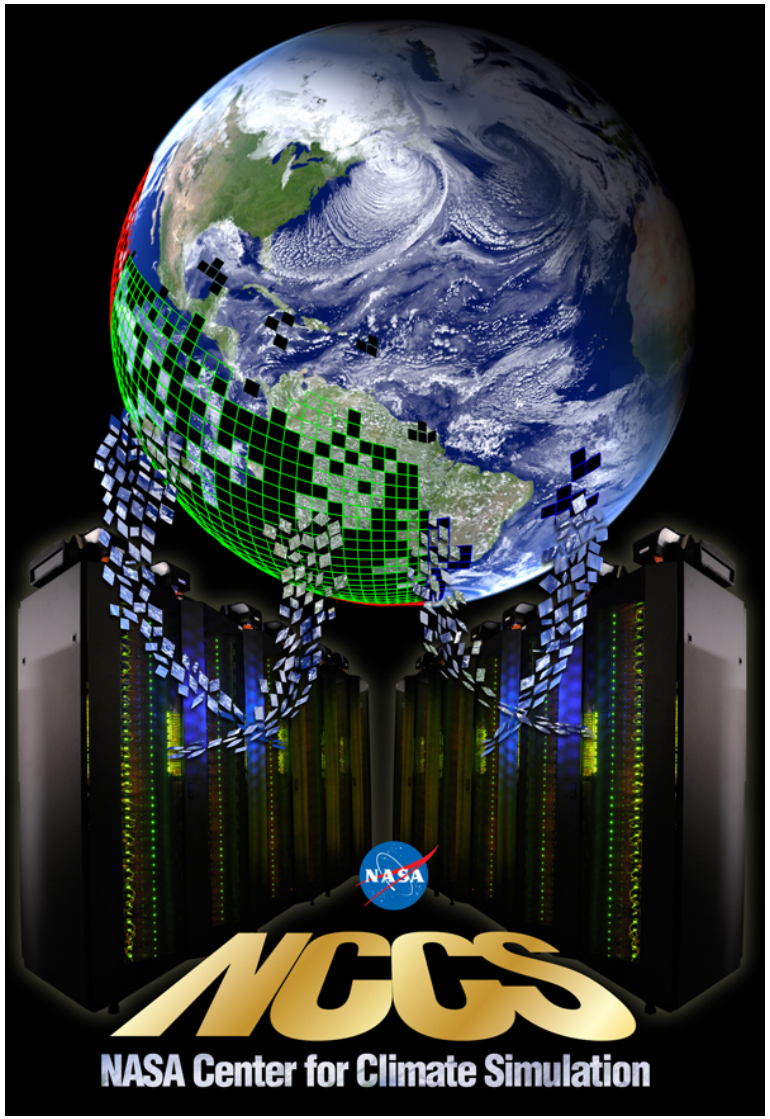
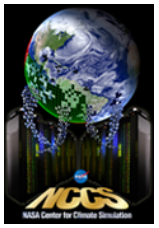


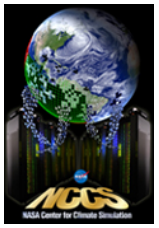
NCCS Brown Bag Series



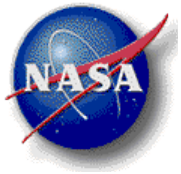
Tips for Monitoring Memory Usage in PBS jobs on Discover

Chongxun (Doris) Pan
doris.pan@nasa.gov
October 16, 2012

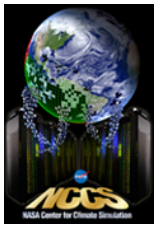
After the talk, you will understand --



- What's memory swapping, really?
- Why does my application cause high memory swapping?
- What could happen when my PBS job causes excessive memory swapping?
- How to determine the memory footprint for your application and decide the appropriate node count
- Tips and Best practices

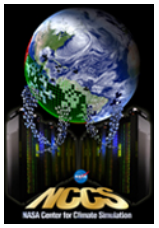


What is memory swapping, really?

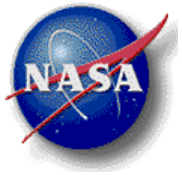


- There is never enough Random Access Memory (RAM)
- Virtual Memory (VM) available = physical RAM + swap space (preconfigured space on the slower hard disk)
- Linux divides VM and physical RAM into chunks of memory, called pages
- Swapping occurs when a page of memory is copied to swap space to free up the page of memory in RAM

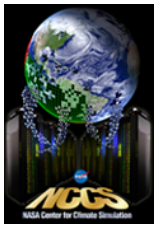
What is memory swapping, really?



- Swapping is necessary because
  1. More memory can be used than is physically available. The kernel is able to swap out less used pages from RAM and free memory for other immediate uses.
  2. Lots of pages used by an application during initialization may not be used until much later.
- Swapping has drawbacks
  - Disks are very slow compared to memory
  - Excessive swapping, or thrashing, occurs where a page is swapped out and then very soon swapped in and then swapped out again, and so on. This is indicative of insufficient RAM for the workload .

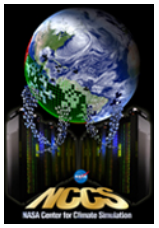


Why does my application cause high "memory swapping"?



- Because the application requires more memory than is physically available on either some or all of the nodes it has been running on
- Things to check:
  - Do I use the correct MPI library?
  - Do I set "mpiprocs" or "-perhost" correctly?
  - How much memory does each MPI processor use?
  - Do some MPI processes require considerably more memory than the rest?

Discover nodes



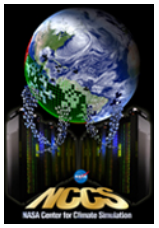
Node	Memory per node	Memory per core	Swap Space per node	Environment
Nehalem	24 GB	3 GB	8 GB	PBS11
Westmere	24 GB	2 GB	8 GB	PBS11
SandyBridge	32 GB	2 GB	8 GB	PBS11
Warp Westmere	48 GB	4 GB	8 GB	PBS11
Dali (01-08)	256 GB	16 GB	--	No PBS
Dali-gpu (09-20)	192 GB	16 GB	--	No PBS

A PBS job will be killed when it uses >=60% of the swap space on one or more nodes.

What could happen when my PBS job swaps excessively?

- Your job is killed when it uses 60% of total 8G swap space on one or more compute nodes
- Thrashing is extremely detrimental to system performance. The nodes may become unresponsive and have to be rebooted
- Sometimes the thrashing happens so fast that it outpaces the system's defensive mechanism and locks up the global file system, which leads to an entire GPFS hang on Discover and Dali
- You will get a few emails from NCCS and receive a call from me…

So, how to prevent that?



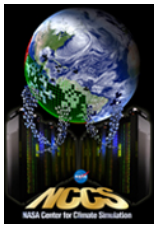
1. The easier way: Run the job interactively and monitor memory usage while it is running
2. Use tools or libraries to obtain memory usage statistics across MPI processors
   - A memory monitoring tool developed by Tyler Simon

   http://www.nccs.nasa.gov/primer/computing.html#memoryreq

   https://modelingguru.nasa.gov/docs/DOC-1727
   - TotalView MemoryScape

   http://www.nccs.nasa.gov/images/Totalview-Part2-Doris.pdf

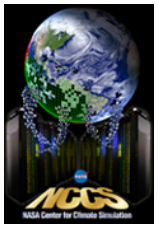
Interactive-batch PBS Jobs



- An interactive-batch job is a regular batch job, but allows you to get on to the compute nodes. Very useful for debugging and computational steering.

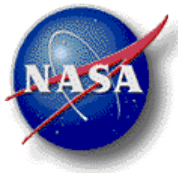
```
$ xsub -I –V -l select=4:ncpus=12:proc=west,walltime=2:00:00 –q general
Establishing X forwarding and submitting batch job...
qsub: waiting for job 845848.pbsa1 to start
qsub: job 845848.pbsa1 ready

borge107:$ xterm &
(Now you are on the headnode. And you can open another terminal!)
borge107:$ cat $PBS_NODEFILE
borge107.prv.cube
borge108.prv.cube
borge118.prv.cube
borge119.prv.cube
borge107:$ mpirun –perhost 6 –np 24 ./GEOSgcm.x
```

Interactive-batch PBS Jobs

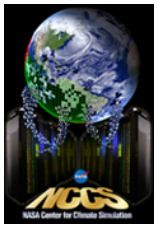


- While the “mpirun” executes, on the other terminal you can issue the "top" command or look at the file /proc/meminfo periodically.

```
borge107:$ top
(Type “q” to quit the top window)
borge107:$ ssh –XY borge119
(You can also ssh to any other nodes listed in your PBS_NODEFILE and
check the status there. Xforwarding is allowed.)
borge119:$ xterm &
borge119:$ top –u cpan2
borge119:$ top –b –n 1 | grep –i GEOSgcm.x
borge119:$ /usr/local/other/Htop/1.0/bin/htop –u cpan2
(Try htop. It is an updated top with more features)
borge119:$ cat /proc/meminfo
```



The “top” command

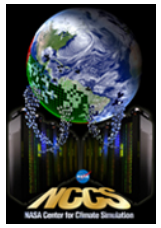


```
top - 10:07:02 up 4 days, 23:26,  1 user,  load average: 10.66, 4.31, 3.37
Tasks: 266 total,  13 running, 252 sleeping,   0 stopped,   1 zombie
Cpu(s): 88.8%us, 10.4%sy,  0.0%ni,  0.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    24022M total,    5748M used,   18274M free,     104M buffers
Swap:    8001M total,      97M used,    7903M free,     476M cached

PID    USER     PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+    COMMAND
29558  cpan2    20   0 4097m 409m  20m R  100  1.7    2:11.80  GEOSgcm.x
29560  cpan2    20   0 3857m 168m  19m R  100  0.7    2:12.03  GEOSgcm.x
29563  cpan2    20   0 4952m 1.2g  21m R  100  5.3    2:09.75  GEOSgcm.x
29565  cpan2    20   0 3856m 166m  18m R   99  0.7    2:11.46  GEOSgcm.x
    1  root     20   0 10392   92   60 S    0  0.0    0:04.62  init
    2  root     20   0     0    0    0 S    0  0.0    0:00.00  kthreadd
```

VIRT: Total amount of virtual memory used by the process, including code, data, shared libraries, pages that swapped out

RES: Resident size (i.e., non-swapped physical memory a process has used)

/proc/meminfo



```
MemTotal:       24599140 kB
MemFree:        18709396 kB
Buffers:          106632 kB
Cached:           434020 kB
SwapCached:        58180 kB
…
SwapTotal:       8193140 kB
SwapFree:        8093128 kB
…
Shmem:             25564 kB
…
VmallocTotal:   34359738367 kB
VmallocUsed:       466688 kB
VmallocChunk:   34344742748 kB
…
```

- /proc/meminfo shows top level memory information.
- /proc/<PID>/status shows information for a given process ID
- The information is dynamic, changing constantly.
- cat /proc/29558/status | egrep 'VmSize|VmRSS'

From the top output or meminfo, once you see that MemFree is down to 1GB and lower, and SwapFree is down to 4GB and still steadily creeping down, kill your mpirun job before it is too late!

The correct way to kill your mpirun job is:
mpirun –perhost 1 –np <#_of_nodes> killall –v <command_name>

Tips and Best Practices

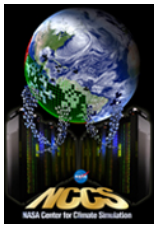
1. If running all cores per node causes memory problems, you should request more nodes and run fewer cores per node.

```
#PBS –l select=6:ncpus=12:mpiprocs=6
…
# if using Intel MPI
mpirun –perhost 6 –np 36 ./foo.exe

# if using MVAPICH2
mpirun –ppn 6 –np 36 ./foo.exe

# If using OpenMPI
mpirun –npernode 6 –np 36 ./foo.exe
```

select=<# of nodes>
mpiprocs=<# of MPI tasks per node you want to run your job with
ncpus=<minimum # of total cores per node for your requested nodes>

e.g., select=6:ncpus=12 may either get you 6 Westmere nodes or 6 SandyBridge nodes, depending on which nodes become available first

Tips and Best Practices



2. Note! One Intel MPI library, mpi/impi-4.0.3.008, has a known problem of ignoring the "mpiprocs" setting. Any other Intel MPI lib is fine.

   So, it is a good practice to always specify "-perhost" (or –ppn, -npernode) for mpirun when you intend to use only part of the total cores per node

Tips and Best Practices

3. Remember: If neither mpiprocs nor –perhost/-ppn/-npernode is specified, the default value is the total available cores on the nodes.

```
#PBS –l select=6:ncpus=8
…
mpirun –np 48 ./foo.exe
```

You may get 6 Nehalem nodes. In this case, the job executes on 6 nodes running 8 processors on each node.

You may get 6 SandyBridge nodes. In this case, the job executes on 3 nodes running 16 processors on each node. The rest of 3 nodes are left idle!!

You may get 6 Westmere nodes. In this case, the job executes on 4 nodes running 12 processors on each node. The rest of 2 Westmere nodes are left idle!!

Tips and Best Practices

4. Adding "proc="may be necessary if you intend to run on a certain type of nodes.

```
#PBS –l select=6:ncpus=8:mpiprocs=8:proc=neha
…
mpirun –np 48 ./foo.exe
```

```
#PBS –l select=4:ncpus=12:mpiprocs=12:proc=west
…
mpirun –np 48 ./foo.exe
```

```
#PBS –l select=3:ncpus=16:mpiprocs=16
…
mpirun –np 48 ./foo.exe
```

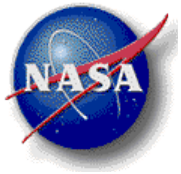
proc=sand is unnecessary in this case because currently the SandyBridge nodes are the only choice satisfying ncpus=16

Tips and Best Practices

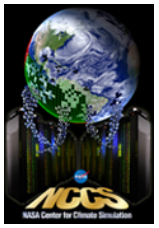
5. Some applications may benefit from uneven placement of MPI processors per node because some processors need access to larger memory than the rest.

```
#PBS –l select=1:ncpus=12:mpiprocs=2+3:ncpus=12:mpiprocs=12
…
# if using Intel MPI, do NOT use impi-4.0.3.008
mpirun –np 36 ./foo.exe
```

Processors: (p0,p1)(p2,p3,…p13)(p14,p15,…p25)(p26,p27,…p35)
Nodes: node1 node2 node3 node4



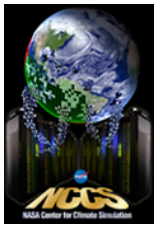
Tips and Best Practices



6. During the interactive session, make sure to clean up hanging processes before issuing the next mpirun command!

```
$ xsub -I -l select=4:ncpus=12:mpiprocs=12:proc=west,walltime=2:00:00 –q
general
…qsub: job xxxxxx.pbsa1 ready

borgd009:$ …
borgd009:$ mpirun –np 48 ./GEOSgcm.x
(Ctl-c to terminate the job may leave many dangling processes both on the head node
and other compute nodes!!)
borgd009:$ mpirun –perhost 1 –np 4 killall –v GEOSgcm.x
(This will kill all the processes running GEOSgcm.x on all 4 nodes)
borgd009:$ mpirun –np 48 ./GEOSgcm.x
(Now you can issue another mpirun command as all 4 nodes are cleared of any leftover
processes from the previous run. Use "top" to verify if you want)
```

Tips and Best Practices



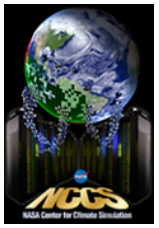
7. Take advantage of the large memory on the Dali nodes. You are not allowed to run multi-processor MPI jobs on Dali, but you can:
   - Monitor memory usage for a small MPI job while running it with a single processor;
   - Monitor memory usage for a single pre- or post-processing job. Use the information later to decide how many instances of similar, independent jobs can execute concurrently on a Discover computer node using either PoDs or other scripts.

Tips and Best Practices

8. If a new job of yours has repeatedly run nodes out of memory no matter how you tweak your script, don't hesitate to contact support@nccs.nasa.gov.

We can help you.

Tips and Best Practices



9. Try NOT to attempt a new memory-intensive application that you know may run nodes out of memory during weekends or nighttime, when the admin team cannot respond quickly to perform damage control on the system